A Self-adaptive Artificial Bee Colony Algorithm with Symmetry Initialization

Yu Xue[1,2,3], Jiongming Jiang[1], Tinghuai Ma[1], Jingfa Liu[1], Wei Pang[4,5]
[1] School of Computer and Software, Nanjing University of Information Science and Technology, China
[2] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, China
[3] Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science & Technology, China
[4] School of Natural and Computing Sciences, University of Aberdeen, UK
[5] Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, China
xueyu@nuist.edu.cn, jiongmingjiang@163.com, {thma, jfliu}@nuist.edu.cn, pang.wei@abdn.ac.uk

Abstract

The Artificial Bee Colony (ABC) algorithm is an optimization algorithm inspired by the foraging behavior of bee swarms. Similar to some evolutionary algorithms, there is a main limitation in ABC, i.e., in many problems, ABC is good at exploration but poor at exploitation. Thus, in order to overcome this limitation and improve the performance of ABC when dealing with various kinds of optimization problems, we proposed a self-adaptive artificial bee colony algorithm with symmetry initialization (SABC-SI). In our SABC-SI algorithm, a novel population initialization method based on half space and symmetry is designed, and such method can increase the diversity of initial solutions. Besides, a self-adaptive search mechanism and several new Candidate Solution Generating Strategies (CSGSes) have also been developed. Consequently, the evolutionary strategies can be selected dynamically according to their search performance. Moreover, the selection operator is improved by eliminating some of the poor solutions and making good use of the two best solutions in both the current and previous generations. The novel algorithm was tested on 25 different benchmark functions. The experimental results show that SABC-SI outperforms several state-of-the-art algorithms, which indicates that it has great potential to be applied to a wide range of optimization problems.

Keywords: Artificial Bee Colony, Population initialization, Selection strategy, Self-adaptive

1 Introduction

Because the evolutionary algorithms (EAs) have few parameters and they can be easily applied to various problems, many researchers have devoted to the study of EAs over the last three decades. Compared with other traditional methods, EAs have become well-established global optimization methods, and they all have high robustness and broad applicability. Possessing the characteristics of self-organizing and self-learning, EAs can deal with those challenging optimization problems which otherwise cannot be effectively solved by traditional optimization methods. The popular EAs, including genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], differential evolution (DE) [3], ant colony optimization (ACO) [4] and artificial bee colony (ABC) [5], have excellent ability to handle many complex real-world problems.

In addition to the above EAs, Karaboga proposed the ABC algorithm in 2005 [15]. The experimental results of ABC on numerical optimization problems indicate that ABC has distinctive characteristics when compared with other traditional optimization algorithms [5].

Due to its simplicity, good efficiency and high robustness, ABC is a frequently used tool for solving many real-world optimization problems [6-9]. However, similar to other EAs, ABC also has its own limitations. Specifically, ABC sometimes cannot find the best solution and thus fall into premature convergence [10]. Therefore, some researchers have proposed several revised algorithms to overcome these disadvantages.

To improve ABC's solution search strategy, which is good at exploration but poor at exploitation, Zhu and Kwong developed the gbest-guided ABC (GABC) by incorporating the information of global best solution into the solution search equation with an aim to improve the exploitation [11]. Although the performance of GABC is enhanced through better use of information about the global best solution, GABC is also sometimes restricted by the current global best solution when solving some optimization problems. To further enhance the global convergence, Gao *et al.* proposed a modified population initialization method called chaotic and opposition-based learning method to

*Corresponding Author: Yu Xue; Email: xueyu@nuist.edu.cn
DOI: 10.3966/160792642018091905007

produce population [12]. But the initial population produced by the improved population method is sometimes redundant in the sense that identical individuals are generated. In addition, in order to use the parameters of ABC during the evolutionary process, Bansal *et al.* proposed an adaptive version of ABC where the step size in solution modification and the ABC parameter *limit* are determined adaptively based on the current fitness values [13]. Interestingly, Semwal proposed a deep learning approach and it described human push recovery data classification through the use of the features from intrinsic mode functions, and such features were obtained through empirical mode decomposition on different leg joint angles [14]. However, the idea of that paper was inspired by the body organs of biology, and the focus of our algorithm is self-adaptive biological behaviors. These are two different research aspects.

In this paper, in order to overcome the limitations and further improve the performance of ABC on optimization problems, we propose a self-adaptive artificial bee colony algorithm with symmetry initialization (SABC-SI), which employs a novel self-adaptive search mechanism as well as a new population initialization strategy. In this new self-adaptive mechanism, several candidate solution generating strategies (CSGSes) can be dynamically selected in each generation according to their search performance. In addition, the novel population initialization method is proposed based on half space and symmetry method, and it can increase the diversity of solutions. The new initialization method can fundamentally guarantees the uniform distribution of the initial population. Furthermore, we also improve the solution selection operator by following the Darwin's theory of evolution, that is, the superior solutions would be more likely to survive and the inferior solutions would be more likely to be eliminated during the evolution process.

The rest of this paper is organized as follows. Section 2 summarizes the ABC algorithm. The self-adaptive artificial bee colony algorithm based on symmetry initialization and optimal solution reserving selection operator is presented and analyzed in Section 3. In Section 4, we report the experimental results, and this is followed by Section 5, which concludes the paper and explores future work.

2 Related Work

2.1 The Artificial Bee Colony Algorithm

The idea of ABC for numerical optimization was first proposed by Karaboga in 2005 [15] and was described in detail in 2007 [5]. There are three different types of artificial bees in ABC: employed bees, onlooker bees, and scout bees. Employed bees are pioneers of the swarm and they have the ability to discover food sources and gather honey. According to the information shared by employed bees, the onlooker bees would choose the better food sources for exploitation. The function of the scout bees is that they abandon bad food sources and search for new ones. Besides, these three kinds of colonies can be transformed into each other and they form an effective co-operative system. The main steps of ABC algorithm are briefly described as follows:

In the initialization phase of ABC, a population (colony) is generated. The number of colony size is equal to the number of employed bees and onlooker bees. Besides, the number of food sources equals the half of the colony size. The initial population is generated by Equation (2.1). Then, the fitness values of these individuals are evaluated.

$$X_{i,j} = Lbound_{i,j} + rand * (Ubound_{i,j} - Lbound_{k,j}), \quad \textbf{(2.1)}$$

where $i = 1, 2, \ldots, \text{ps}$, and ps is the number of individuals in the population; $j = 1, 2, \ldots, \text{D}$, and D is the number of dimensions of the problem; *rand* is a random number between 0 and 1, *Lbound* and *Ubound* are the lower and upper bounds of a problem, respectively.

After initialization, the ABC algorithm enters an iterative process in the employed bee phase. A new individual is generated as follows:

$$X_{i,j}^{G+1} = X_{i,j}^{G} + rand(X_{i,j}^{G} - X_{k,j}^{G}), \quad \textbf{(2.2)}$$

where G represents the number of current generation and G+1 is the next generation. K is a random integer in the range $[1, 2, \ldots, ps]$, $j = 1, 2, \ldots, \text{D}$, $rand \in [-1,1]$. $X_{i,j}^{G}$ is the current individual and $X_{i,j}^{G+1}$ is the generated individual.

If a generated value is out of the boundary, it will be shifted back to the boundary. Then, the fitness value of the new solution is calculated. If the new fitness value is better than the old one, the new solution would replace the previous one and the trial counter will be reset. Otherwise, the previous solution is reserved and its trial counter is increased.

Before the onlooker bee stage, the selection probabilities of the candidate solutions are calculated through Equation (2.3). The candidate solutions with higher probabilities will be further optimized. Then, for the chosen individual, a new individual is generated by Equation (2.2) and its fitness value is calculated. If the new fitness value is better than the old one, the new solution would replace the previous one and the trial counter is reset. Otherwise, the previous solution is reserved and its trial counter is increased.

$$P_i = \frac{fit_i}{\sum_{k=1}^{ps} fit_k}, \quad \textbf{(2.3)}$$

where fit_i represents the fitness value of solution i and ps is the number of individuals. P_i is the selection probability of the current solution.

In the scout bee stage, each individual has a trial counter. The solutions which could not be improved within a given number of trials are abandoned. If the solution whose trial counter exceeds the limitation, it would be abandoned and a new solution would be generated by Equation (2.4). Again, the correlative trial counter is reset.

$$X_{i,j} = X_{min,j} + rand(X_{max,j} - X_{min,j}), \quad \textbf{(2.4)}$$

where $i = 1, 2, \ldots, ps$, $j = 1, 2, \ldots, D$, $rand \in [0,1]$, X_{max} and X_{min} are the lower and upper bounds of problems.

After the above iterative process, all candidate solutions are generated. The *ith* optimal solution is found, whose fitness value is the best. Algorithm 1 shows the detailed procedures of the original ABC.

Algorithm 1. The main steps of the ABC algorithm

Step (1) Initialization
  Step (1.1) Generate ps individuals form the initial population $\mathbf{X} = \{x_1, x_2, ..., x_{SN}\}$ randomly
  Step (1.2) Calculate the fitness values of the ps individuals
  Step (1.3) Initialize trial, limit, D and maxCycle
Step (2) Employed Bee Stage
  Step (2.1) For (i=1, i<ps, i++)…, ps, CONTINUE
    Step (2.1.1) Use Formula (2.2) to generate a new solution
    Step (2.1.2) Calculate fitness value of the new solution
  Step (2.2) If find the better value, update the solution and trial=0, else trial=trial+1
  End For
Step (3) Calculate probability values by Formula (2.3)
Step (4) Onlooker Bee Stage
  Step (4.1) If rand < $\mathbf{P}_i$
    Step (4.1.1) Use Formula (2.2) to generate a new solution
    Step (4.1.2) Calculate fitness value of the new solution
    Step (4.1.3) If find the better value, update the solution and trial=0, else trial=trial+1
  End If
Step (5) Scout Bee Stage
  If max(trial)>limit, produce new individuals through Formula (2.4)
Step (6) Generate and record the best solution
Step (7) If iterations>= maxCycle, output the optimal solution, else go to Step2

2.2 The Improved ABC

The ABC algorithm is a simple, effective, and robust evolutionary optimization method. Compared with the other EAs, the ABC algorithm is relatively young yet more efficient. Many researchers are interested in the ABC algorithm, because it has fewer control parameters and it is easier to use [5]. ABC has been widely applied to many real-world problems. For example, ABC has been applied to a loudspeaker design problem [16], and for the design of two-channel quadrature mirror filter banks [17]. Besides, ABC was used to minimize the makespan for the single machine batch processing problem when the sizes of the jobs are non-identical [18]. Horng also adopted ABC for a stochastic economic lot scheduling problem [19], and Pan solved the large-scale hybrid flow shop scheduling problem with ABC [20]. What is more, ABC has also been employed to solve an interest-based forwarding problem [21].

As a result, ABC has emerged as a powerful tool for solving both local and global optimization problems [10-11]. However, there is no specific algorithm that can achieve the best solution for all optimization problems, the so-called no free lunch theorem. The ABC algorithm also has some disadvantages. For instance, it may sometimes stop approaching to the global optimum (or optima) even before the population converging to a local optimum, and it may suffer from extensive computation time due to its stochastic nature. Hence, it is necessary to continue to improve the efficiency of ABC when solving many problems. Gao and Liu proposed a chaotic and opposition-based learning method to produce initial population [22]. In this method, the initial population is first generated through the chaotic method. Then, the opposition-based learning method is used to generate another population. It is worth to notice that the size of the former population is the same as the latter. Next, the fitness values of the two populations are evaluated.

Half of the whole populations whose fitness values are better than the others are reserved and they become the final initial population. Experimental results show that the ABC algorithm employing this initialization approach outperforms the original ABC algorithm [22]. Meanwhile, we also investigate an optimal solution reserving selection operator to make the full use of the previous optimum generated in each circulation. Besides, we construct an adaptive strategy selection pool. One strategy could be chosen at a high probability during the evolutionary process. Then the strategy with the higher probability is more likely to be selected than the others and this could greatly improve the ability of solving different kinds of problems.

3 Self-adaptive Artificial Bee Colony Algorithm Based on Symmetry Initialization and Optimal Solution Reserving Selection Operator

3.1 The Symmetry Initialization Method Based on Half Space

Population initialization is an important phase in EAs. A good initial population tends to lead to good solutions and accelerates the convergence speed. As mentioned in Section 2.2, a chaotic and opposition-based learning method which is designed by Gao and Liu is used to produce the initial population [22]. The detailed steps are as follows. First, the chaotic method is used to generate the initial population. Next, another population is produced by the opposition-based learning method. The size of the two populations is the same. Then, their fitness values are calculated. The individuals are further compared with each other and the better ones are reserved. Finally, the final initial population is obtained. Experimental results show that this initialization approach is effective [22].

However, this initialization approach also has several disadvantages: the individuals in the population which are generated by the chaotic system may be the same as those in the population which are generated through the opposition-based learning method. In this case, the efficiency will be affected. As shown in Figure 1(a), suppose the range of a given problem is $[X_1, X_2]$ and the generated population is $\{R_1, R_2, R_3, R_4\}$, which are produced by the chaotic method, where $X_1, X_2 \in R$, $M = (X_1+X_2)/2$. Obviously, the values generated through the opposition-based learning method are still $\{R_1, R_2, R_3, R_4\}$.

This means that the opposition-based learning method is invalid in this case. As a result, almost half of the initial population are redundant and its efficiency is thus affected in the algorithm.

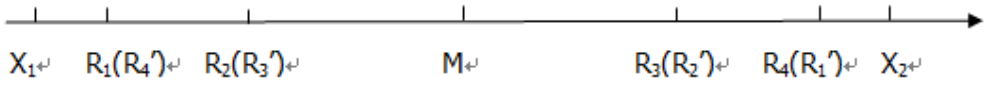
In order to overcome the above shortcoming, we propose a novel initialization approach named Half Space based Symmetry Initialization algorithm (HSSI). The detailed steps are as follows: The first and utmost step is to define the Half Range (HR) of a given problem and it is selected randomly. It is noted that the selection of HR is a further research problem. In this research we choose the front half range. After that the first part of the initial population is generated in HR randomly, the symmetry initialization approach is then used to produce the second part of the initial population in the rest of the range. Finally, the final population is generated by selecting the top half individuals with better fitness values from both populations. As shown in Figure 1(b), this method divides the solution space into N regions and each region is assigned to a solution every time by HSSI. In this research we set N=2. R_1, R_2, R_3 and R_4 are produced from $[X_1, M]$. R_1', R_2', R_3', and R_4' are produced by the symmetry method. It essentially guarantees the uniform distribution of initialization. Obviously, the initial population produced by this approach is more comprehensive and efficient. The detailed steps of the HSSI algorithm are shown in Algorithm 2.

3.2 The Optimal Solution Reserving Selection Operator

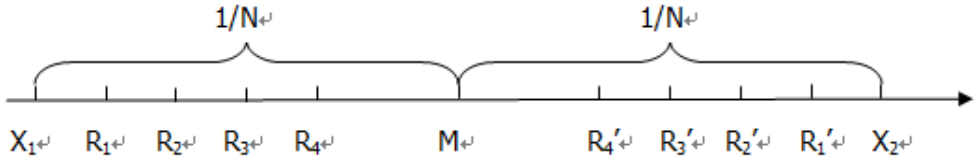
During the evolution process in each generation, employed bees, onlooker bees and scout bees are employed alternatively. Thereafter, the best solution is recorded. For the last iteration of the loop, surely we only need to record the optimal solution. However, we may also need to deal with the other best solutions. Therefore, we consider how to make a better use of the previous optima if it is replaced by the current best solution. So we propose a selection operator which can researve optimal solutions.

After the scout bee phase of the original ABC algorithm, the solutions of the current generation are generated. Then, the current optimum (*bestFit*) is compared with the previous optimum (*gbestFit*). If *bestFit* is better than *gbestFit*, the current best solution (*bestSol*) is retained. Otherwise, this step is skipped. Nevertheless, not only does the proposed selection operator complete the above steps, but also it is capable of abandoning bad solutions. Abandoning a few solutions is a direction in future research, and in this research we only consider the case of the worst solution. First, we compare the *gbestFit* with the current worst fitness (*worstFit*). If the *worstFit* is worse than the *gbestFit*, the worst solution is replaced by a random generated solution. Otherwise, this step is skipped. The advantage of this approach is that it can further enhance the diversity of the population and avoid getting trapped into local optima when solving the complex optimization problems. The fundamental reason is that those solutions which seldom or never evolve should be removed or die, which is known as "natural selection and survival of the fittest". Besides,

the selection probability of the good population is increased and the superior population can also improve the overall efficiency of algorithms. The detailed steps of the optimal solutions reserving selection operator are shown in Algorithm 3.



(a)



(b)

Figure 1. A comparisons between HSSI and rand initialization

Algorithm 2. The detailed steps of the HSSI algorithm

Step (1) Input: Population size *ps*; Number of dimension *D*; *M* is a certain value in HR
Step (2) For i = 1 to *ps*
  For j = 1 to *D*
    P(i, j) = *Lbound(j)* + rand(0,1)*(*M* − *Lbound(j)*)
  End for
End for
Step (3) For i = 1 to *ps*
  For j = 1 to *D*
    Q(i, j) = *Lbound(j)* + *Ubound(j)* − P(i, j)
  End for
End for
Step (4) Select *SN* fittest individuals from collection $\{P \cup Q\}$ as the actual initiali population ***PQ***.

Algorithm 3. The optimal solution reserving selection operator

Step (1) Find the best fitness bestFit and the worst fitness worstFit respectively, gbest represents the previous optimum, gbestFit represents the fitness of gbest
Step (2) According to bestFit and worstFit, find the best solution bestSol and the worst solution worstSol and its position worstPos
Step (3) If bestFit < gbestFit
  gbest = bestSol
  gbestFit = bestFit
End if
Step (4) If worstFit > gbestFit
  randomSol = Lbound + rand(Ubound − Lbound)
  $\mathbf{population}_{worstPos}$ = randomSol
End if

3.3 The Self-adaptive Mechanism

In ABC algorithm, the employed bee phase and onlooker bee phase both need Candidate Solution Generating Strategies (CSGSes) to produce new individuals. An ideal CSGS should be able to not only solve different problems as many as possible, but also to be adapted to the global and local optimization during the evolutionary process. Many researchers developed various CSGSes and many of them have achieved good results [12, 23-25]. Obviously, one CSGS cannot satisfy the demand of solving various kinds of complex problems. For this reason, a self-adaptive mechanism, combined with several CSGSes, is employed in our SABC-SI. The self-adaptive mechanism is described as follows:

At the initialization phase of the algorithm, each CSGS is assigned with the same initial selection probability value (***P***) and it is set as the reciprocal of the total CSGS. The number of adaptive CSGSes (*straNum*) is set to 8 since eight CSGSes are used in this paper, and the details of these eight CSGSes are presented in Section 3.4. *LP* is the cycle for adaptive update and it is set to 10. First, according to the initial ***P***, each individual is given its own CSGS through the roulette wheel selection algorithm, which is selected from the eight CSGSes. The chosen strategy in the current generation is named *curStra*. Then, the new individual is produced through *curStra* and its fitness value is calculated. If the obtained fitness is better than the previous one, the strategy-success flag matrix (***straFlagS***) of *curStra* is updated. Otherwise, the strategy-failure flag matrix (***straFlagF***) is updated. The ***straFlagS*** and ***straFlagF*** of the other CSGSes are all set to 0 and it represents that these strategies are not selected in the current generation. After one generation, which means that all individuals have been evolved at least once, the data of ***straFlagS*** and ***straFlagF*** are transferred to the total-success flag matrix (***totalFlagS***) and total-failure flag matrix (***totalFlagF***), respectively. Meanwhile, ***straFlagS*** and ***straFlagF*** are set to zero in order to prepare for the next generation. When the number of above iterations reaches *LP*, the new selection probabilities of each CSGS will be calculated according to Equations (3.1) and (3.2).

$$P_q^{'}=\begin{cases}\dfrac{\sum_{n=1}^{LP} straFlagS_n}{(\sum_{n=1}^{LP} straFlagS_n+\sum_{n=1}^{LP} straFlagF_n)}, & \sum_{n=1}^{LP} straFlagS_n \neq 0\\ \dfrac{\sum_{n=1}^{LP} straFlagS_n}{(\varepsilon+\sum_{n=1}^{LP} straFlagF_n)}, & \sum_{n=1}^{LP} straFlagS_n = 0\end{cases}, \quad \textbf{(3.1)}$$

where ***straFlagS*** represents the strategy-success flag matrix and ***straFlagF*** represents the strategy-failure flag matrix. $\sum_{n=1}^{LP} straFlagS_n$ and $\sum_{n=1}^{LP} straFlagF_n$ represent the number of successful and failed strategies during the evolutionary process, respectively. $P_q^{'}$ represents the probability of successful strategies. ε is a small number close to 0.

$$P = P_q^{'} / \sum_{q=1}^{Q} P_q^{'}, \quad \textbf{(3.2)}$$

where $q \in \{1,2,...,Q\}$, and ***P*** is the next selection probabilities of strategies.

$$X_{i,j}^{G+1} = X_{r1,j}^{G} + rand(X_{r2,j}^{G} - X_{r3,j}^{G}) + rand(X_{r4,j}^{G} - X_{r5,j}^{G}), \quad \textbf{(3.3)}$$

$$X_{i,j}^{G+1} = X_{i,j}^{G} + rand(X_{best,j}^{G} - X_{i,j}^{G}) + rand(X_{r1,j}^{G} - X_{r2,j}^{G}), \quad \textbf{(3.4)}$$

$$X_{i,j}^{G+1} = X_{i,j}^{G} + rand(X_{best,j}^{G} - X_{i,j}^{G}) + rand(X_{r1,j}^{G} - X_{r2,j}^{G}) + rand(X_{r3,j}^{G} - X_{r4,j}^{G}), \quad \textbf{(3.5)}$$

where G represents the number of current generation and G+1 is the next generation, r_1, r_2, r_3 and r_4 are random integers in the range of $[1, 2, \ldots, ps]$, $j = 1, 2, \ldots, D$, $rand \in [-1,1]$. $X_{i,j}^{G}$ is the current individual and $X_{i,j}^{G+1}$ is the generated individual.

The detailed steps of SABC-SI are described in Algorithm 4.

3.4 The Candidate Solution Generating Strategy

The CSGS greatly affects the efficiency of the SABC-SI algorithm. In general, the efficiency is better when the number of adaptive CSGS is bigger, but more strategies mean that the algorithm is more complex and the running time may be longer. After carefully weighing these factors, the *straNum* is set at 8. In order to avoid getting trapped into local optima, Zhu added the perturbation of global optimal solution and proposed GABC algorithm [11]. Furthermore, Gao enhanced the effect of best solution in the current generation and presented the ABC/best/1, ABC/best/2 and MABC algorithms [12, 22]. Besides, Gao also proposed ABC/rand/1 algorithm for increasing the diversity of solution and the trial vectors are all generated from stochastic solutions [25].

In order to solve multimodal problems by better approaches and make the strategy more comprehensive, inspired by the mutation strategies of DE [23-24], we proposed three mutation strategies for ABC, as listed by Equations (3.3), (3.4) and (3.5). They are termed as ABC/rand/2, ABC/current-to-best/1 and ABC/current-to-best/2.

The above 8 variant strategies form an adaptive strategy selection pool. According to the strategy selection probability ***P***, one strategy could be chosen at a high probability during the evolutionary process. Then the strategy with a higher probability is more likely to be selected than the others, and this could greatly improve the ability of SABC-SI for solving different kinds of problems.

Algorithm 4. The proposed SABC-SI algorithm

Step (1) Initialization
  Step (1.1) Initialize the strategy selection probability $\mathbf{P}_{\mathbf{straNum}}$(q= 1, 2, ..., Q), where Q is the number of strategies, and straNum is the number of CSGSes; set the values of ps, LP, **straFlagS**, **straFlagF**, **totalFlagS** and **totalFlagF**.
  Step (1.2) Algorithm 2 is used to generate ps individuals
  Step (1.3) Calculate the fitness values of the ps individuals
Step (2) Employed Bee Phase
  Step (2.1) For (fitCount< MaxFES fitCounti++), 1, …, ps, CONTINUE
    Step (2.1.1) Based on the related data gathered from evolution, the current strategy curStra can be selected by the roulette wheel algorithm.
    Step (2.1.2) Use curStra from the candidate pool to generate a new solution and calculate the fitness value of the new solution.
  Step (2.2) If a the better value is found, update the solution, else trial=trial+1
  Step (2.3) Update the flag matrices **straFlagS** and **straFlagF**.
Step (3) Update **totalFlagS** and **totalFlagF**
Step (4) Calculate the probability values
Step (5) Onlooker Bee Phase
  Step (5.1) if rand< $\mathbf{P}_i$
    Step (5.1.1) Use curStra from the candidate pool to generate a new solution and calculate the fitness value of the new solution.
    Step (5.1.2) If a better fitness value is found, update the solution, else trial=trial+1
    Step (5.1.3) Update the flag matrices **straFlagS** and **straFlagF**.
  End if
  End for
Step (6) Update **totalFlagS** and **totalFlagF**
Step (7) Scout Bee Phase
  If max(trial)>limit, generate one individual randomly
Step (8) If the number of iterations for adaptive update achieves LP, calculate the new $\mathbf{P}_{\mathbf{straNum}}$
Step (9) Use Algorithm 3 to generate best solutions and remove the worst previous solution
Step (10) If FES>= MaxFES, output the optimal solution, else go to step2

4 Computational Experiments and Result Comparisons

4.1 The Orthogonal Design Method

In this paper, the orthogonal design method is used to find out the best level of the main influential factors of a specific algorithm. The Population Size (PS), *limit*, and the fixed number *LP* are analyzed in this research. Each factor is set with three levels according to experience and they are shown in Table 1.

Table 1. The three factors and their levels

Factors	Levels		
PS	60	120	180
Limit	500	1000	1500
LP	10	20	30

In order to find the best level for each factor, we carry out a series of experiments for the combinations of all factor levels and select the best combination. In this experiment, the number of combinations is 3*3*3=27. In general, we consider a situation where there exists A factors and B levels and the number of combinations is A*B. Obviously, when A, and B are very large, the combinatorial experiments for all factors may be intractable. Thus, we expect to provide a design method which can obtain the good combination based on fewer experiments. The orthogonal design is a very popular method which can deal with this problem. As shown in Table 2, 9 orthogonal tests are designed according to the orthogonal array in this paper.

Table 2. Orthogonal design

Combinations	PS	Limit	LP	Results
M_1	60	500	10	f_1
M_2	60	1000	20	f_2
M_3	60	1500	30	f_3
M_4	120	500	20	f_4
M_5	120	1000	30	f_5
M_6	120	1500	10	f_6
M_7	180	500	30	f_7
M_8	180	1000	10	f_8
M_9	180	1500	20	f_9

4.2 The Benchmark Functions and Parameter Settings

The factor analysis on the 25 benchmark functions is recorded. Due to space limitation, we only list the data on function 1 as shown in Table 3. The optimal numbers of each level of each factor are shown in Table 4. Based on these results, we find that the best PS is 60 and it is exactly equal to the double of D. The best *limit* is 500 and it is approximately 10 times bigger than PS. The best *LP* is 10 or 20 and it tends to be smaller.

Table 3. Factor analysis on function 1

L_1	$(f_1+f_2+f_3)/3$=7.58E-14	$(f_1+f_4+f_7)/3$=6.95E-14	$(f_1+f_6+f_8)/3$=8.84E-14
L_2	$(f_4+f_5+f_6)/3$=6.95E-14	$(f_2+f_5+f_8)/3$=8.84E-14	$(f_2+f_4+f_9)/3$=8.21E-14
L_3	$(f_7+f_8+f_9)/3$=1.01E-13	$(f_3+f_6+f_9)/3$=8.84E-14	$(f_3+f_5+f_7)/3$=7.58E-14
Sort	120	500	30

Table 4. The results on the 25 test functions

Parameter	Optimal Number
PS_{60}	18
PS_{120}	12
PS_{180}	8
$Limit_{500}$	18
$Limit_{1000}$	9
$Limit_{1500}$	8
LP_{10}	13
LP_{20}	13
LP_{30}	11

In the experiments, the number of dimensions *D* of the solution is set to 30 or 50. The particle numbers *ps* is equal to the double of D. That is to say, the number of employed bees is half of the population size. To make a fair comparison, the populations for these dimensions of all the compared algorithms were initialized with the same random seeds. The fixed integer of previous generations *LP*=10. The number of decision variables was set to the same for all the 25 test functions. For each algorithm on each function, 25 independent runs were conducted with 500,000 (Number of Fitness Evaluation, *NFE*) times of function evaluations being the termination criterion.

All the algorithms involved in this research are tested on 25 benchmark functions. The individual functions are shown in Table 5 and Table 6. The functions are numbered from f_1 to f_{25} and they can be grouped into three categories: f_1 to f_5 are unimodal functions, f_6 to f_{14} are basic multimodal functions, and f_{15} to f_{25} are composition functions. A detailed description of these test functions can be found in [26].

4.3 Performance Comparison between the HSSI Method and Random Initialization

In order to investigate whether the convergence rate is greatly influenced by the quality of initial population in ABC and HSSI is better than random initialization, the performance of ABC-HSSI (ABC combines only with the HSSI algorithm) on 25 benchmark functions is compared with that of ABC using random initialization. Figure 2 shows the convergence performance of both ABC-HSSI and ABC with random initialization. In order to make the images clearer, we convert *generation* and *fitness* to log(*generation*) and *relevantFitness* respectively. So the horizontal axis and vertical axis represent the corresponding evolution generations and relevant fitness, respectively. The test functions of Figure 2 are from the three different types with different dimensions, and due to space limitation the other results are shown in the supplementary material.

Table 5. Benchmark functions f_1 to f_{20}

Objection function	Property Class	Search Range
$f_1(X)=\sum_{i=1}^{D} Z_i^2 + f\_bias_1$	Unimodal	$[-100,100]^D$
$f_2(X)=\sum_{i=1}^{D}(\sum_{j=1}^{i} Z_j)^2 + f\_bias_2$	Unimodal	$[-100,100]^D$
$f_3(X)=\sum_{i=1}^{\mathrm{D}}(10^6)^{\frac{i-1}{D-1}} Z_i^2 + f\_bias_3$	Unimodal	$[-100,100]^D$
$f_4(X)=(\sum_{i=1}^{D}(\sum_{j=1}^{i} Z_j)^2)*(1+0.4\lvert N(0,1)\rvert) + f\_bias_4$	Unimodal	$[-100,100]^D$
$f_5(X)=max\{\lvert A_i x - B_i\rvert\} + f\_bias_5$	Unimodal	$[-100,100]^D$
$f_6(X)=\sum_{i=1}^{D-1}(100(Z_i^2 - Z_{i+1})^2 + (Z_i - 1)^2) + f\_bias_6$	Basic Multimodal	$[-100,100]^D$
$f_7(X)=\sum_{i=1}^{D-1}\frac{Z_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{Z_i}{\sqrt{i}}) + 1 + f\_bias_7$	Basic Multimodal	$[0,600]^D$
$f_8(X)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} Z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi Z_i)) + 20 + e + f\_bias_8$	Basic Multimodal	$[-32,32]^D$
$f_9(X)=\sum_{i=1}^{D}(Z_i^2 - 10\cos(2\pi Z_i) + 10) + f\_bias_9$	Basic Multimodal	$[-5,5]^D$
$f_{10}(X)=\sum_{i=1}^{D}(Z_i^2 - 10\cos(2\pi Z_i) + 10) + f\_bias_{10}$	Basic Multimodal	$[-5,5]^D$
$f_{11}(X)=\sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k (Z_i + 0.5))]) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k \cdot 0.5)] + f\_bias_{11}$	Basic Multimodal	$[-0.5,0.5]^D$
$f_{12}(X)=\sum_{i=1}^{D}(\mathrm{A}_i - \mathrm{B}_i(x))^2 + f\_bias_{12}$	Basic Multimodal	$[-\pi,\pi]^D$
$f_{13}(X)=F8(F2(Z_1,Z_2)) + F8(F2(Z_2,Z_3)) + ... + F8(F2(Z_D,Z_1)) + f\_bias_{13}$	Basic Multimodal	$[-3,1]^D$
$f_{14}(X)=EF(Z_1,Z_2,...,Z_D) = F(Z_1,Z_2) + F(Z_2,Z_3) + ... + F(Z_{D-1},Z_D) + F(Z_D,Z_1) + f\_bias_{14}$	Basic Multimodal	$[-100,100]^D$
f_{15} : $f_{1-2}(x)$: Rastrigin's Function $f_{3-4}(x)$: Weierstrass Function $f_{5-6}(x)$: Griewank's Function $f_{7-8}(x)$: Ackley's Function $f_{9-10}(x)$: Sphere Function	Composition	$[-5,5]^D$
f_{16} : All settings are the same as f_{15} except $\mathbf{M_i}$	Composition	$[-5,5]^D$
$f_{17}(X)=G(x)*(1+0.2\mid N(0,1)\mid) + f\_bias_{17}$	Composition	$[-5,5]^D$
f_{18} : $f_{1-2}(x)$: *Ackley's* Function $f_{3-4}(x)$: *Rastrigin's* Function $f_{5-6}(x)$: *Sphere* Function $f_{7-8}(x)$: *Weierstrass* Function $f_{9-10}(x)$: *Griewank's* Function	Composition	$[-5,5]^D$
f_{19} : All settings are the same as f_{18} exceptσ and λ	Composition	$[-5,5]^D$
f_{20} : All settings are the same as f_{18} except $\boldsymbol{O}$	Composition	$[-5,5]^D$

Table 6. Benchmark functions f_{21} to f_{25}

Objection function	Objection function	Objection function
f_{21} : $f_{1-2}(x)$: *Rotated Expanded* Scaffer's F6 Function $f_{3-4}(x)$: *Rastrigin's* Function $f_{5-6}(x)$: *F8F2* Function $f_{7-8}(x)$: *Weierstrass* Function $f_{9-10}(x)$: *Griewank's* Function	Composition	$[-5,5]^D$
f_{22} : All settings are the same as f_{21} except $\mathbf{M_i}$	Composition	$[-5,5]^D$
f_{23} : All settings are the same as f_{21} except $\mathbf{x_j}$ and round	Composition	$[-5,5]^D$
f_{24} : $f_1(x)$: *Weierstrass* Function $f_2(x)$: Rotated Expanded Scaffer's F6 Function $f_3(x)$: F8F2 Function $f_4(x)$: Ackley's Function $f_5(x)$: Rastrigin's Function $f_6(x)$: Griewank's Function $f_7(x)$: Non-Continuous Expanded Scaffer's F6 Function $f_8(x)$: Non-Continuous Rastrigin's Function $f_9(x)$: High Conditioned Elliptic Function $f_{10}(x)$: Sphere Function with Noise in Fitness	Composition	$[-5,5]^D$
f_{25} : All settings are the same as f_{24} except no exact search range set for this test function.	Composition	$[2,5]^D$

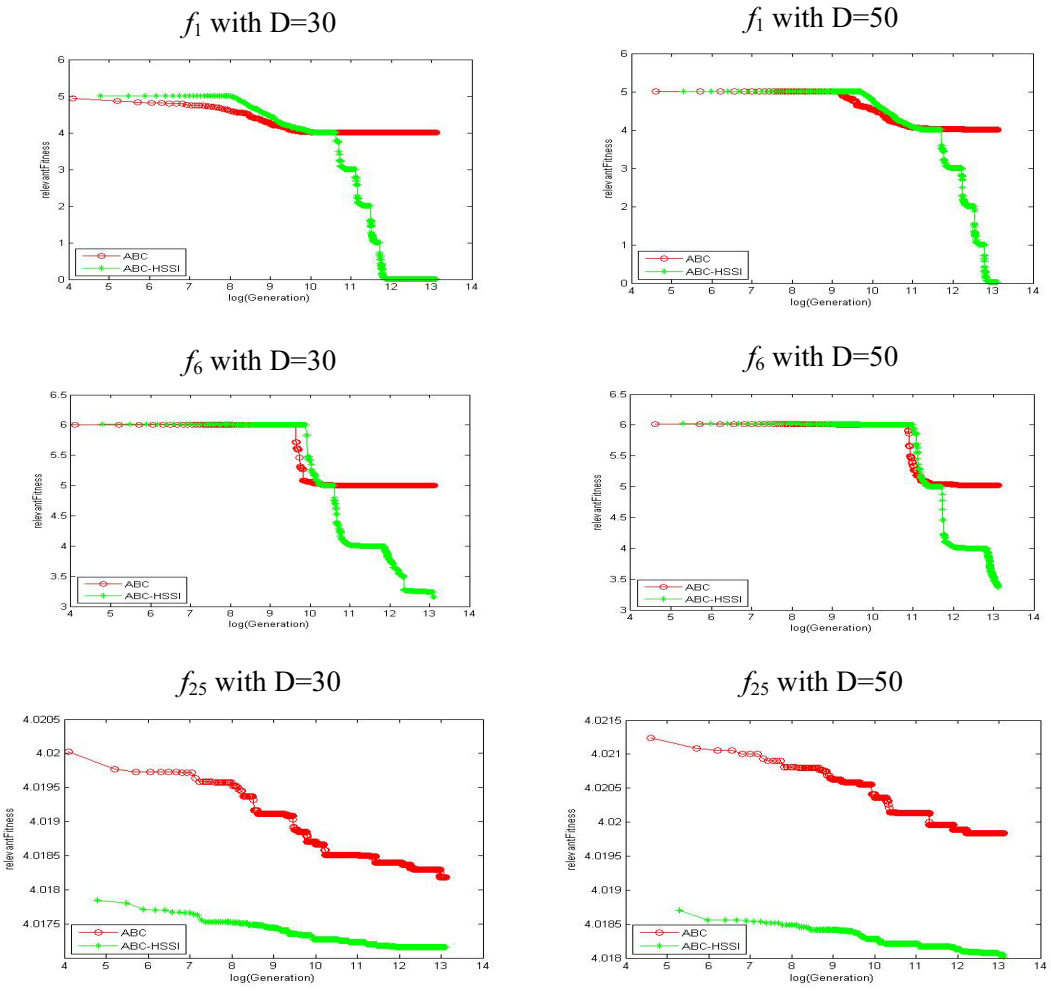


Figure 2. Convergence performance of ABC-HSSI and ABC

As shown in Figure 2, the convergence rate of ABC-HSSI is better than ABC with random initialization on f_1, f_6, and f_{25}. Although the performance of ABC-HSSI is a little poorer in the middle generations of the whole evolution process for f_1, the results of ABC-HSSI are superior to the other one obviously. The case is similar on other test functions which are not shown in Figure 2. It benefits most from the HSSI and symmetric method which make the search more global-oriented and the initial population is distributed more uniformly. Obviously, the algorithm with a randomly initialized population could not get the results which are as good as ABC-HSSI. To sum up, the HSSI approach can produce better initial population and it is helpful for ABC to obtain better solutions.

4.4 The Effect of the Optimal Solution Reserving Selection Operator

To investigate the effectiveness of the novel solution selection operator, the performance on the 25 benchmark functions of ABC-MS (ABC combined only with the modified optimal solution reserving selection operator) is compared with ABC with the original selection strategy. Figure 3 shows the convergence performance of ABC-MS and ABC. Same as above, the test functions of Figure 3 are also from three different types of different dimension and the results on other test functions are shown in the supplementary material.

It can be observed from Figure 3 that the convergence performance of ABC-MS is better than ABC with the original selection operator when solving different kinds of benchmark functions.

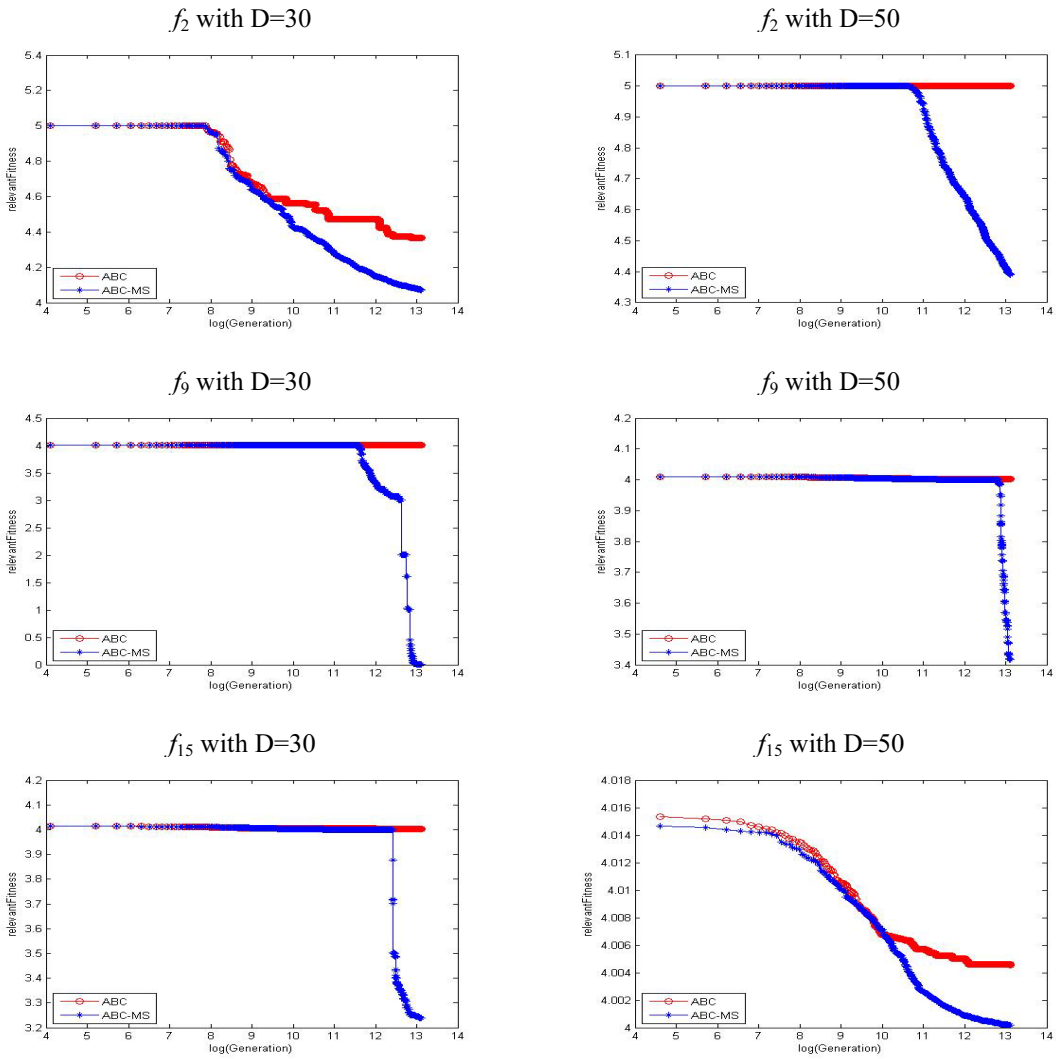


Figure 3. Convergence performance of ABC-MS and ABC

Table 7. The optimization results of GABC, ABC/best/1, NSHS and SABC-SI on 25 test functions with D=30

Functions	GABC		ABC/best/1		NSHS		SABC-SI	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_1	1.09E-02	6.33E-03	1.97E-12	1.93E-13	2.42E-02	3.71E-03	**5.68E-14**	0.00E+00
f_2	2.99E+04	6.02E+03	3.55E+04	1.96E+03	3.16E+03	1.03E+03	**3.10E+03**	1.53E+03
f_3	4.88E+07	7.10E+06	5.28E+07	9.41E+06	8.39E+06	3.92E+06	**1.31E+06**	3.04E+06
f_4	4.60E+04	1.95E+03	5.06E+04	5.02E+03	9.47E+03	1.59E+03	**3.08E+03**	1.61E+03
f_5	1.09E+04	2.90E+02	9.34E+03	5.34E+02	**2.64E+03**	4.72E+02	6.36E+03	7.31E+02
f_6	1.44E+03	7.37E+02	1.86E+02	4.11E+01	2.22E+02	9.14E+01	**4.04E-01**	3.26E-01
f_7	4.70E+03	8.97E-01	4.70E+03	5.15E-02	3.96E+02	8.75E+00	**3.52E+03**	5.25E-13
f_8	**2.09E+01**	9.03E-02	**2.09E+01**	1.10E-02	2.10E+01	2.59E-02	**2.09E+01**	3.34E-02
f_9	3.64E+01	4.88E+00	2.03E-12	2.02E-12	7.07E-01	9.18E-01	**5.68E-14**	0.00E+00
f_{10}	3.11E+02	9.17E+00	2.82E+02	1.14E+01	1.90E+02	8.94E+00	**1.45E+02**	1.47E+01
f_{11}	3.36E+01	3.98E-01	3.32E+01	1.45E+00	4.03E+01	6.19E-01	**2.50E+01**	1.16E+00
f_{12}	8.10E+04	2.49E+04	8.82E+04	1.61E+04	9.22E+03	3.18E+03	**7.84E+03**	2.92E+03
f_{13}	7.58E+00	4.75E-01	7.59E+00	7.45E-01	2.24E+00	2.59E-01	**9.69E-01**	9.99E-02
f_{14}	1.32E+01	1.04E-01	1.33E+01	3.39E-02	1.36E+01	6.68E-02	**1.26E+01**	7.88E-02
f_{15}	3.36E+02	1.19E+02	3.29E+02	1.45E+02	3.08E+02	8.66E-01	**1.63E+02**	2.31E+02
f_{16}	3.66E+02	4.10E+01	3.17E+02	1.11E+01	2.98E+02	7.86E+01	**2.63E+02**	1.34E+02
f_{17}	4.65E+02	2.36E+01	4.25E+02	8.83E+00	3.62E+02	5.82E+01	**2.89E+02**	2.69E+01
f_{18}	9.24E+02	2.32E+00	9.21E+02	2.16E+00	8.70E+02	2.84E+00	**8.58E+02**	2.06E+00
f_{19}	9.32E+02	1.76E+00	9.21E+02	1.73E+00	8.98E+02	1.75E+00	**8.64E+02**	1.64E+00
f_{20}	9.27E+02	9.21E-01	9.23E+02	3.07E+00	8.67E+02	2.40E+00	**8.50E+02**	2.93E-01
f_{21}	5.14E+02	1.62E+00	5.07E+02	5.87E-01	**5.00E+02**	9.01E-04	**5.00E+02**	8.63E-13
f_{22}	1.07E+03	1.04E+01	1.04E+03	1.60E+01	**9.37E+02**	1.70E+00	**9.37E+02**	7.09E+00
f_{23}	6.99E+02	1.49E+00	6.12E+02	2.13E+01	5.34E+02	1.50E-04	**5.12E+02**	3.38E-04
f_{24}	1.18E+03	1.79E+01	1.07E+03	3.25E+01	2.00E+02	4.05E-04	**1.01E+02**	0.00E+00
f_{25}	1.71E+03	3.65E+00	1.68E+03	5.07E+00	1.64E+03	2.20E+00	**1.63E+03**	3.93E+00

"Mean" and "SD" refer to the average values and standard deviation of the corresponding functions respectively.

Table 8. The optimization results of GABC, ABC/best/1, NSHS and SABC-SI on 25 test functions with D=50

Functions	GABC		ABC/best/1		NSHS		SABC-SI	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_1	1.08E+00	7.42E-01	6.33E-11	1.10E-11	2.58E-01	4.77E-02	**1.71E-13**	4.64E-14
f_2	9.92E+04	3.74E+03	1.06E+05	1.22E+04	4.32E+04	8.78E+03	**3.92E+04**	4.22E+03
f_3	2.20E+08	1.68E+07	1.84E+08	4.77E+07	6.08E+07	3.39E+07	**2.70E+07**	6.45E+06
f_4	1.48E+05	1.68E+04	1.31E+05	7.70E+03	7.69E+04	1.15E+04	**6.32E+04**	1.46E+04
f_5	2.64E+04	6.50E+02	2.47E+04	1.20E+03	**6.08E+03**	9.11E+02	6.84E+03	1.92E+03
f_6	2.19E+04	2.09E+03	3.24E+03	1.33E+03	4.70E+02	1.05E+02	**6.83E-01**	1.22E-01
f_7	6.24E+03	1.39E+01	6.22E+03	1.85E+00	6.20E+03	3.01E+01	**3.55E+03**	1.29E-12
f_8	**2.11E+01**	9.00E-03	**2.11E+01**	1.82E-02	2.12E+01	1.34E-02	**2.11E+01**	6.88E-02
f_9	1.10E+02	9.04E+00	2.97E-12	6.18E-13	6.03E+00	5.28E-01	**2.08E-13**	2.68E-14
f_{10}	8.28E+02	1.14E+01	6.61E+02	1.17E+01	3.58E+02	8.52E+00	**3.40E+02**	8.47E+01
f_{11}	6.39E+01	6.58E-01	6.46E+01	6.63E-01	7.48E+01	1.61E+00	**5.33E+01**	3.85E+00
f_{12}	4.65E+05	8.09E+04	4.69E+05	4.79E+03	4.45E+04	2.20E+04	**3.26E+04**	2.32E+03
f_{13}	2.12E+01	1.75E+00	1.82E+01	1.90E+00	5.09E+00	7.52E-02	**2.58E+00**	6.77E-02
f_{14}	2.30E+01	2.00E-01	2.31E+01	2.34E-01	2.34E+01	6.74E-03	**2.24E+01**	4.16E-01
f_{15}	3.93E+02	6.88E+01	3.17E+02	8.96E+01	3.43E+02	4.07E+01	**1.87E+02**	3.81E+01
f_{16}	4.26E+02	3.91E-01	4.27E+02	1.37E+00	2.65E+02	1.08E+01	**2.63E+02**	1.14E+01
f_{17}	8.52E+02	3.51E+01	7.56E+02	3.98E+01	3.80E+02	6.43E+01	**3.75E+02**	2.69E+01
f_{18}	9.89E+02	2.07E+01	9.71E+02	9.33E+00	9.65E+02	1.41E+01	**9.29E+02**	1.86E+01
f_{19}	9.91E+02	8.55E+00	9.68E+02	1.30E+00	9.70E+02	3.40E+00	**9.22E+02**	1.99E+01
f_{20}	9.91E+02	7.74E+00	9.75E+02	3.36E+00	9.42E+02	1.33E+01	**9.37E+02**	2.07E+01
f_{21}	1.04E+03	1.86E+00	1.04E+03	1.16E+00	**5.00E+02**	3.38E-03	**5.00E+02**	7.03E-13
f_{22}	1.20E+03	5.75E+00	1.13E+03	2.46E+01	9.44E+02	1.32E+01	**9.43E+02**	1.94E+01
f_{23}	1.06E+03	3.30E+00	1.04E+03	3.69E+00	5.65E+02	4.55E-03	**5.39E+02**	1.88E-04
f_{24}	1.40E+03	1.46E+01	1.36E+03	5.29E+00	2.00E+02	3.02E-03	**1.17E+02**	9.62E+01
f_{25}	1.90E+03	4.63E+00	1.85E+03	9.93E+00	1.67E+03	5.52E+00	**1.65E+03**	9.32E+00

"Mean" and "SD" refer to the average values and standard deviation of the corresponding functions respectively.

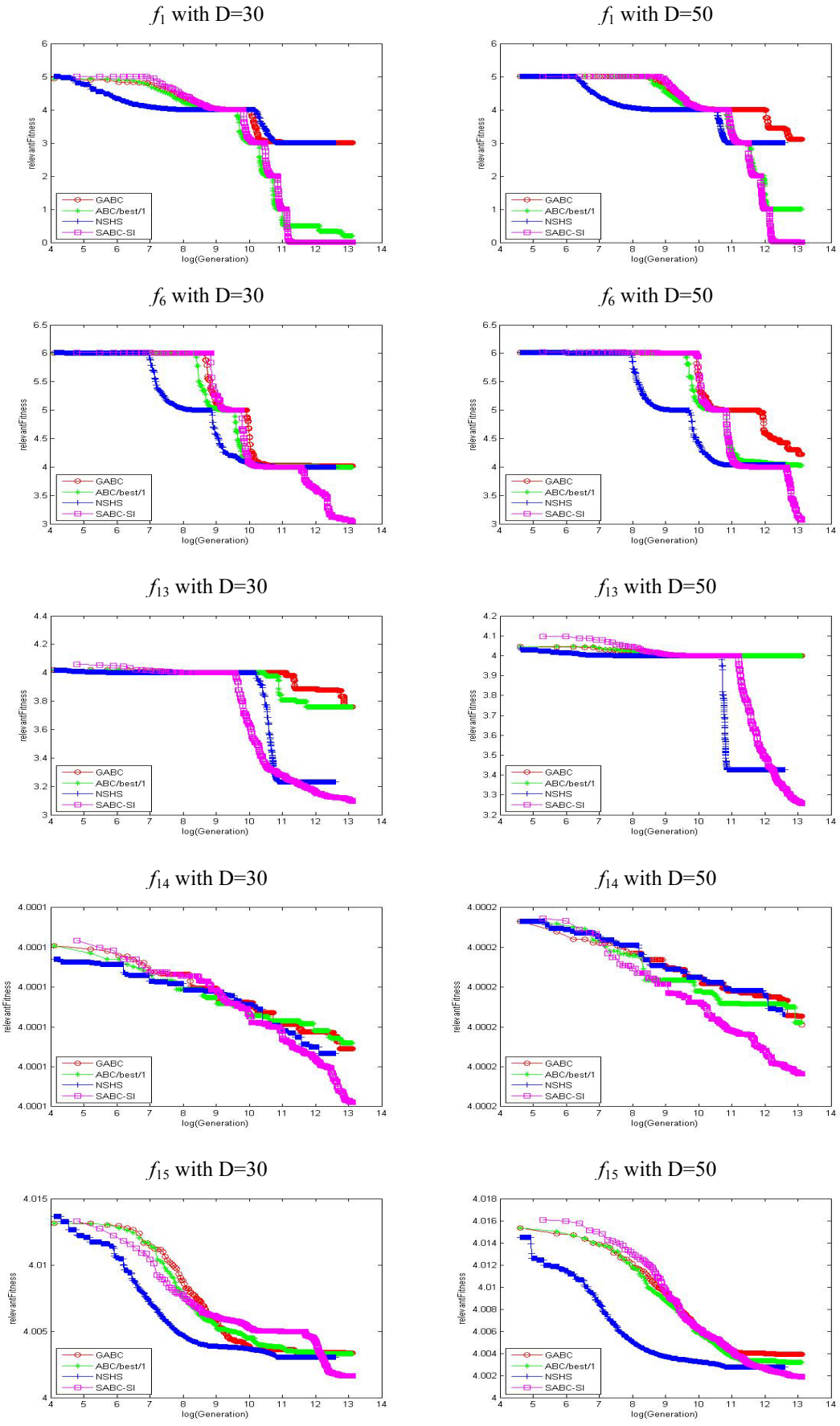


Figure 4. Convergence performance of GABC, ABC/best/1, NSHS and SABC-SI

Albeit the quality of the solutions produced by ABC-MS are close to that of the others at the beginning of the evolution, the optimal solution reserving selection operator plays an important role in the later stage and the convergent rate is greatly accelerated. The final results of ABC-MS are better than those of the other. Thus, we can draw a conclusion that at least the optimal solution reserving selection operator can improve the efficiency of ABC and get better quality solutions.

4.5 The Performance Comparison between SABC-SI and Other Algorithms

To verify the effectiveness and superiority of the proposed algorithm, the performance of SABC-SI is compared with other state-of-the-art algorithms, including GABC [11], NSHS [27] and ABC/best/1 [12]. These algorithms are applied to solve unimodal, multimodal, and composite benchmark functions in this research. Table 7 and Table 8 show the mean and SD of the computational results on different dimensions, respectively. On the whole, the solutions of all functions on all dimensions generated by SABC-SI are better than the others except for f_5. The reason is that SABC-SI employs the adaptive strategy selection mechanism and each function can be greatly solved by the best strategy. In addition, SABC-SI could find the most accurate solutions when solving f_1 and f_9. Meanwhile, we can also observe that GABC, NSHS and ABC/best/1 cannot find the precise solutions on f_1 and f_9. This means that these two types of problems cannot be easily solved by the above algorithms but they can be perfectly settled by SABC-SI. It is interesting to see that not only does SABC-SI discover the best solution for f_8, but also it is found by GABC and ABC/best/1. Both SABC-SI and NSHS find the same solution on all dimensions when solving f_{21}. As shown in Table 7 and Table 8, the solutions of f_{15} to f_{25} which belong to composition functions are not satisfactory. In order to meet the future need of EAs, it is required to study and solve the class of problems more effectively. Overall, we can draw a conclusion that SABC-SI is better than the other algorithms in terms of the precision of solutions.

The convergence performance of the search processes when solving the three different types of functions is shown in Figure 4. The convergence performance of SABC-SI is similar to that of ABC/best/1 on f_1 when D=30, but SABC-SI outperforms ABC/best/1 significantly on f_1 when D=50. Although the efficiency of SABC-SI is comparable to NSHS at the beginning of f_{13}, the latter easily falls into local optimum and NSHS is better than the other algorithms significantly at the end of evolution. The convergence performance of SABC-SI is better than the others through the observation and analysis of these graphs shown in Figure 4 and the results demonstrate that the modified strategies are very effective.

5 Conclusion & Future Work

In this research we proposed a novel artificial bee colony algorithm SABC-SI. SABC-SI employs an improved population initialization method called HSSI, which fundamentally guarantees the uniform distribution of initialization. In addition, the optimal solution reserving selection operator follows the Darwin's theory and the solutions of higher quality can be evolved through this selection operator. Moreover, we also develop a novel self-adaptive mechanism to choose candidate strategies automatically. All of the above enables SABC-SI to effectively deal with different kinds of optimization problems. Experimental results on 25 different benchmark functions demonstrate that SABC-SI performs better than several state-of-the-art algorithms. Hence, SABC-SI is a promising algorithm which can effectively deal with different kinds of optimization problems.

However, there is still room for improvement for SABC-SI. It can be seen from Table 7 that the quality of the solution of SABC-SI on f_5 is not the best. So SABC-SI is still worth to be further studied in serval aspects, such as the selection of HR and appropriately abandoning solutions in the solution selection strategy. These will become our future work. In addition, the practical applications of the proposed approach to a wider range of problem domains will also be further investigated, for instance, in the future we will consider exploring the potential of SABC-SI on a number of data classification [28-29], clustering [30-31], and other applications [32-34]. In particular, we will be very interested in studying how the proposed novel selection operator and population initialization method affect the performance of ABC when solving the above problems.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (Grant No. 61403206), the Natural Science Foundation of Jiangsu Province (Grant No. BK20141005) and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant No. 14KJB520025), the Priority Academic Program Development of Jiangsu Higher Education Institutions, the Research Foundation of Nanjing University of Information Science and Technology (2013x034). WP is supported by the 2016 Scottish Crucible Award funded by the Royal Society of Edinburgh. This work is supported by the Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing (Contract No. SKL2017CP01).

Reference

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control & Artificial Intelligence*, University of Michigan Press, 1975.

[2] J. Kennedy, R. Eberhart, Particle Swarm Optimization, *IEEE International Conference on Neural Networks,* Perth, WA, Australia, 1995, pp. 1942-1948.

[3] R. Storn, K. Price, Differential Evolution: A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341-359, December, 1997.

[4] M. Dorigo, L.-M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53-66, April, 1997.

[5] D. Karaboga, B. Basturk, A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, *Journal of Global Optimization*, Vol. 39, No. 3, pp. 459-471, November, 2017.

[6] X.-Z. Wen, L. Shao, W. Fang, Y. Xue, Efficient Feature Selection and Classification for Vehicle Detection, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 25, No. 3, pp. 508-517, March, 2015.

[7] S.-C. Horng, Combining Artificial Bee Colony with Ordinal Optimization for Stochastic Economic Lot Scheduling Problem, *IEEE Transactions on Systems Man & Cybernetics Systems*, Vol. 45, No. 3, pp. 373-384, March, 2015.

[8] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications, *Artificial Intelligence Review*, Vol. 42, No. 1, pp. 21-57, June, 2014.

[9] B. Gu, V. Sheng, Feasibility and Finite Convergence Analysis for Accurate On-line V-support Vector Machine, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, No. 8, pp. 1304-1315, August, 2013.

[10] D. Karaboga, B. Akay, A Comparative Study of Artificial Bee Colony Algorithm, *Applied Mathematics and Computation*, Vol. 214, No. 1, pp. 108-132, August, 2009.

[11] G.-P. Zhu, S. Kwong, Gbest-guided Artificial Bee Colony Algorithm for Numerical Function Optimization, *Applied Mathematics and Computation*, Vol. 217, No. 7, pp. 3166-3173, December, 2010.

[12] W.-F. Gao, S.-Y. Liu, L.-L. Huang, A Global Best Artificial Bee Colony Algorithm for Global Optimization, *Journal of Computational and Applied Mathematics*, Vol. 236, No. 11, 2012, pp. 2741-2753, May, 2012.

[13] J.-C. Bansal, H. Sharma, K. V. Arya, K. Deep, M. Pant, Self-adaptive Artificial Bee Colony, Optimization, *A Journal of Mathematical Programming and Operations Research*, Vol. 63, No. 10, pp. 1513-1532, May, 2014.

[14] V.-B. Semwal, K. Mondal, G. C. Nandi, Robust and Accurate Feature Selection for Humanoid Push Recovery and Classification: Deep Learning Approach, *Neural Computing & Applications*, Vol. 28, No. 3, pp. 565-574, March, 2017.

[15] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report-TR06, October, 2005,

[16] X. Zhang, X. Zhang, S. L. Ho, W. N. Fu, A Modification of Artificial Bee Colony Algorithm Applied to Loudspeaker Design Problem, *IEEE Transactions on Magnetics*, Vol. 50, No. 2, pp. 737-740, February, 2014.

[17] S. K. Agrawal, O. P. Sahu, Artificial Bee Colony Algorithm to Design Two-channel Quadrature Mirror Filter Banks, *Swarm and Evolutionary Computation*, Vol. 21, pp. 24-31, April, 2015.

[18] M. Al-Salamah, Constrained Binary Artificial Bee Colony to Minimize the Makespan for Single Machine Batch Processing with Non-identical Job Sizes, *Applied Soft Computing*, Vol. 29, No. C, pp. 379-385, April, 2015.

[19] S.-C. Horng, Combining Artificial Bee Colony with Ordinal Optimization for Stochastic Economic Lot Scheduling Problem, *IEEE Transactions on Systems, Man, and Cybernetics-Systems*, Vol. 45, No. 3, pp. 373-384, March, 2015.

[20] J.-Q. Li, Q.-K. Pan, Solving the Large-scale Hybrid Flow Shop Scheduling Problem with Limited Buffers by A Hybrid Artificial Bee Colony Algorithm, *Information Sciences*, Vol. 316, No. C, pp. 487-502, September, 2015.

[21] F. Xia, L. Liu, J. Li, A. M. Ahmed, L. T. Yang, J. Ma, Beeinfo: Interest-Based Forwarding Using Artificial Bee Colony for Socially Aware Networking, *IEEE Transactions on Vehicular Technology*, Vol. 64, No. 3, pp. 1188-1200, March, 2015.

[22] W.-F. Gao, S.-Y. Liu, A Modified Artificial Bee Colony Algorithm, *Computers & Operations Research*, Vol. 39, No. 3, pp. 687-697, March, 2012.

[23] A. K. Qin, P. N. Suganthan, Self-adaptive Differential Evolution Algorithm for Numerical Optimization, *The 2005 IEEE Congress on Evolutionary Computation,* Edinburgh, Scotland, UK, 2009, pp. 1785 -1791.

[24] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, M. N. Vrahatis, Enhancing Differential Evolution Utilizing Proximity-based Mutation Operators, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, pp. 99 -119, February, 2011.

[25] W.-F. Gao, S.-Y. Liu, Improved Artificial Bee Colony Algorithm for Global Optimization, *Information Processing Letters*, Vol. 111, No. 17, pp. 871-882, September, 2011.

[26] Suganthan, Hansen, Liang, Deb, Chen, Auger and Tiwari, *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*, Technical Report, Nanyang Technological University, Singapore And KanGAL Report Number 2005005, May, 2005.

[27] K.-P. Luo, A Novel Self-adaptive Harmony Search Algorithm, *Journal of Applied Mathematics*, Vol. 2013, pp. 1-16, 2013.

[28] B. Gu, V. S. Sheng, A Robust Regularization Path Algorithm for V-Support Vector Classification, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 5, pp. 1241-1248, May, 2017.

[29] X.-Z. Wen, L. Shao, Y. Xue, W. Fang, A Rapid Learning Algorithm for Vehicle Classification, *Information Sciences*, Vol. 295, No. C, pp. 395-406, February, 2015.

[30] J.-C. Ji, W. Pang, Y.-L. Zheng, Z. Wang, Z.-Q. Ma, A Novel Artificial Bee Colony Based Clustering Algorithm for Categorical Data, *Plos One*, Vol. 10, No. 5, e0127125, May, 2015. doi:10.1371/journal.pone.0127125

[31] Y.-H. Zheng, B. Jeon, D. Xu, J. Wu, H. Zhang, Image Segmentation by Generalized Hierarchical Fuzzy C-means Algorithm, *Journal of Intelligent and Fuzzy Systems*, Vol. 28, No. 2, pp. 961-973, March, 2015.

[32] B. Gu, V. S. Sheng, K.-Y. Tay, W. Romano, S. Li, Incremental Support Vector Learning for Ordinal Regression, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 7, pp. 1403-1416, July, 2015.

[33] C. Zheng, H. E., M. Song, J. Song, Optimal Crowdsourcing Scheme for Open Questions on Web, *Journal of Internet Technology*, Vol. 17, No. 5, pp. 897-904, September, 2016.

[34] T.-Y. Ma, P. Gerber, A Hybrid Learning Algorithm for Generating Multi-Agent Daily Activity Plans, *Journal of Internet Technology*, Vol. 17, No. 5, pp. 959-969, September, 2016.

Biographies

Yu Xue is a member of IEEE (92058890), ACM (2270255), and CCF (E200029023M). He received the Ph.D. degree from College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, China, in 2013. He is a lecturer in the School of Computer and Software, Nanjing University of Information Science and Technology.

Jiongming Jiang received the undergraduate degree from the School of Computer and Software, Nanjing University of Information Science and Technology. He is currently a master student in the School of Computer and Software, Nanjing University of Information Science and Technology, China. His research include biomimetic algorithm.

Tinghuai Ma is a professor in Computer Sciences at Nanjing University of Information Science & Technology, China. He received his Bachelor (HUST, China, 1997), Master (HUST, China, 2000), Ph.D. (Chinese Academy of Science, 2003) and was Post-doctoral associate (AJOU University, 2004). His research interest is data mining.

Jingfa Liu received the B.Sc. degree in mathematics from Hunan Normal University, Changsha, China, in 1995. The M.Sc. degree in operational research and cybernetics from Shanghai Railway University, Shanghai, China, in 1999. The Ph.D. degree in computer software and theory from Huazhong University of Science and Technology, Wuhan, China.

Wei Pang received his Ph.D. degree in computing science from the University of Aberdeen in 2009. He is currently a lecturer at the University of Aberdeen. His research interests include bio-inspired computing, data mining, machine learning, and qualitative reasoning. He has published over 60 papers, including 30+ journal papers.